Is the Genetic Algorithm a Cooperative Learner?

Helen G. Cobb
Navy Center for Applied Research in Artificial Intelligence
Naval Research Laboratory, Code 5514
Washington, D. C. 20375-5320
Internet: cobb@aic.nrl.navy.mil

Abstract

This paper begins to explore an analogy between the usual competitive learning metaphor presented in the genetic algorithm (GA) literature and the cooperative learning metaphor discussed by Clearwater, Huberman, and Hogg. In a blackboard cooperative learning paradigm, agents share partial results with one another through a common blackboard. By occasionally accessing the blackboard for a partial solution, an agent can dramatically increase its speed in finding the overall solution to a problem. The study of Clearwater et al. shows that the resulting speed distribution among the agents is lognormal. The GA can also be described in terms of an analogous cooperative learning paradigm. Unlike the blackboard learner, the GA shares information by copying and recombining the solutions of the agents. This method of communication slows down the propagation of useful information to agents. The slower propagation of information is necessary because the GA cannot directly evaluate parts of a solution or "partial solutions." The extent to which the GA is cooperative also depends on the choice of heuristics used to modify the canonical GA. The few test cases presented in this paper suggest that the GA may at times yield an approximately lognormal distribution or a mixture of lognormal distributions. While the results look promising, more analysis of the algorithm's overall process is required.

1 INTRODUCTION

Cooperation is generally recognized as an important underlying principle of our social systems. A natural extension of this principle is the application of cooperation in computational systems. Along with the increasing interest in developing parallel architectures, there has also been a corresponding increase in studying distributed problem solving (e.g., Bond and Gasser, 1988). Many artificial intelligence systems illustrate the benefits of cooperation. For example, the HEARSAY-II speech understanding system is one of the more noted systems (Fennell and Lesser, 1975).

The recent work by Clearwater et al. (Clearwater, Huberman, and Hogg, 1991; Huberman, 1990; Huberman and Hogg, 1987) examines the quantitative value of cooperative learning from both a theoretical and an empirical point of view. One of their papers emphasizes the importance and generality of their theoretical results (Clearwater, Huberman, and Hogg, 1991, pp. 2):

> It showed that cooperative searches, when sufficiently large, can display universal characteristics, independent of the detailed nature of either the individual processes or the particular problem being tackled. This universality manifests itself in two separate ways. First, the existence of a sharp transition from exponential to polynomial time required to find the solution as heuristic effectiveness improved [Huberman and Hogg, 1987]. Second, the appearance of a lognormal distribution in [an] individual agent's problem solving effectiveness. The enhanced tail of this distribution guarantees the existence of some agents with superior performance. This can bring about a combinatorial implosion [Kornfeld, 1982] with superlinear speed-up in the time to find the answer with respect to the number of processes.

The classical framework for analyzing generational genetic algorithm (GA) (Holland, 1975; De Jong, 1975) tends to emphasize the role of competition in the "survival of the fittest." Competing structures within the population explore various parts of the search space in parallel. The important ability of the GA to evaluate several schemata at once within the GA's population without "explicit bookkeeping" is called *implicit parallelism* (Holland, 1975; Goldberg, 1989). During each generation, the algorithm selects parts of the search space to explore: first, by applying the selection and reproduction operators to generate clones of relatively more fit structures, then by applying the crossover operators to generate new structures from substructures of the clones (i.e., possible *building blocks*), and finally, by mutating these new structures by a small amount to ensure some variety in the population. For function optimization problems, the competition over the generations evolves a population of relatively homogeneous structures. These structures provide an optimal and/or a near-optimal solution to the problem.

What if, instead of emphasizing the notion of competition, we describe the GA in a complementary way? Suppose we regard the GA population as a set of *cooperating agents*, where each agent has a trial solution to the problem. The selection function, based on environmental evaluations, effectively determines which solutions are to be shared among the agents. During reproduction, better performing agents provide their solutions to poorly performing agents. Agents also communicate parts of their solutions, i.e., "partial solutions," to other agents during crossover. Agents survive, but their trial solutions may

change with each generation, depending on the information received from other agents and the amount of individual search each agent performs. In the standard GA, which uses haploid structures, each agent stores only one solution at a time.

If the generational GA does fall under the umbrella of the general framework called cooperative learning, especially when applied to problem solving and function optimization, then this additional perspective should enhance our understanding of the algorithm. Clearly, the GA is a cooperative learner in that it shares information among population members during reproduction and crossover, but perhaps a more important question is: "To what extent is the genetic algorithm a cooperative learner?"

The discussion above suggests a parallel between the competitive learning framework usually adopted when analyzing GAs and the cooperative learning framework discussed by Clearwater et al. The GA can be viewed from two complementary perspectives: competitive learning and cooperative learning. This paper further explores the relationship between the two perspectives within the standard GA. The paper then provides a brief analysis of the GA in terms of cooperative learning. A few empirical examples of GA performance illustrate the degree to which the GA is a cooperative learner. The empirical results indicates that although the GA is essentially cooperative, there are some significant differences between the cooperative learner described by Clearwater et al. and the GA. The Conclusions section addresses possible explanations for these differences.

2 OVERVIEW OF THE PROBLEM

This section first explains the assumptions behind blackboard cooperative learning and then compares these assumptions to the GA. Next, the section discusses the blackboard model of cooperative learning as a multiplicative process. A similar analysis is then applied to the Schema Theorem in GAs. This analysis shows that the GA is multiplicative with respect to the growth and decay of a schema; however, a schema may initially alternate in its growth and decay from one generation to the next, depending on the context of the current set of solutions in the population.

2.1 BLACKBOARD COOPERATIVE LEARNING

In the framework of cooperation in distributed problem solving, the blackboard learning system described by Clearwater et al. represents a simple form of cooperation called *results-sharing* (Smith and Davis, 1988). Smith and Davis describe results sharing among *knowledge sources* (KSs) as follows (Smith and Davis, 1988, pp. 66): "Results-sharing is a form of cooperation in which individual nodes assist each other by sharing partial results, based on somewhat different perspectives on the overall problem." In the blackboard learner, a KS is an agent that either searches independently for a solution, or posts and/or reads *hints* from a shared blackboard. If on the average, the hints are beneficial, then the synergetic effect of the interactions among the agents can have a substantial effect in reducing the search time of an agent. In effect, an agent enhances its performance by treating other agents as explicit (but fallible) teachers. If the search heuristics of the agents are effective, i.e., there is a high rate of posting good hints on the blackboard, then the reduction in an agent's search time can be dramatic.[1]

A hint does not have to be part of the final solution; a hint may only be correct in a limited context. In fact, if an agent were to occasionally receive a bad or misleading hint, the overall performance of the system would be unaffected so long as, on the average, hints are good. The value of a hint is not fixed. The same hint may be valued differently by different agents, depending on the processing context of the particular agent. Also, as the search progresses, the value of a partial solution tends to lessen. As more partial solutions are placed on the blackboard, an agent is more likely to have already tested a posted partial solution in a number of contexts. The rate of placing hints on the blackboard is determined by an agent's ability to generate partial solutions (and in realistic systems, the willingness of the agent to communicate them). In the blackboard cooperative learning framework, the agent is allowed to reconsider hints; however, an agent does not reexamine a possible overall solution more than once.

2.2 BLACKBOARD COOPERATIVE LEARNING AND THE GA

If we adopt the cooperative learning framework in thinking about the GA, then a generation is one of many repeated encounters among agents sharing solutions, and the population structures are the trial solutions of the agents that are formed by accepting, to a greater or lesser degree, the hints of other agents. Substructures within the structures, or schemata, represent partial solutions. The overall effect of applying the selection, reproduction and crossover operators to the population is analogous to selectively trading hints on a finite capacity blackboard. During each generation, an agent can respond in four different ways. An agent can: (1) maintain the same solution from the last generation, (2) adopt the solution of another agent, (3) modify its solution from the last generation by accepting hints from another agent, and (4) adopt another agent's solution and then modify that solution with hints from yet another agent. The initial diversity in the population aids individuals in their search; the mutation operator provides each agent with individual learning capability.[2] Each agent performs independent search to the extent that the pseudo-random number generator is uncorrelated.

When we examine the performance of both types of learners, there appears to be a number of similarities. The analysis provided by Clearwater et al. considers the trade-off faced by each agent between *cooperative learning* and the exploration of the search space. In the GA literature, this trade-off is usually expressed as *exploitation* versus exploration (Holland, 1975). Regardless of the terminology, placing extreme emphasis on cooperation (or

---

1. Depending on the quality of the heuristics, cooperation has a beneficial effect even when there are only a few agents involved. For example, a recent study of Clouse and Utgoff (Clouse and Utgoff, 1992) illustrates how a human expert's knowledge (i.e., a teacher having a "perfect heuristic") can accelerate the learning of a reinforcement learner. In the study, the expert (the first agent) periodically supplies actions to be taken (partial solutions) to the reinforcement learner (the second agent). Even with infrequently supplied additional expert knowledge, the reinforcement learner can learn in up to two orders of magnitude fewer trials.

2. Clearly, simple mutation represents the least directed of possible search strategies that an agent can pursue as an individual. The advantages of using hybrid approaches that combine a GA with gradient search strategies is currently a topic of ongoing debate (e.g., Ackley, 1991).

similarly exploitation) to the exclusion of exploration, or *vice versa*, is suboptimal. Let us consider the two cases. First, let us suppose that all of the agents dedicate their time to only seeking the partial solutions of other agents so that there is very little investment in individual search. This means that the agents can only form new hypotheses based on the combination of partial solutions already present in the population. If the number of agents is small, and if there is too much sharing of information among too few agents, it may not be possible for the agents to find a solution. More information can be exchanged by increasing the size of the population and/or by having agents perform more individual search. The same problem arises in the GA when using a high crossover rate with a harsh selection policy, a small population size, and a small mutation rate. These parameter settings emphasize exploitation in the GA's search while limiting the amount of exploration. In the worst case, the GA would not find the solution due to premature convergence. As the other extreme case, let us suppose that all of the agents in the blackboard cooperative learner work separately, without sharing information. In this case, the time required for an agent to find a solution may be very long, depending on the talent of the individual agent. The same problem occurs in the GA when using a large mutation rate, a weak selection policy, and no crossover. In the worst case, the GA would not converge.

Despite these overall similarities in behavior, there are some significant operational differences between the models as outlined below:

- In the blackboard model, each agent knows the correctness of a partial solution (though it may not be part of the optimal solution), whereas in the GA, the agent only knows the value of an overall solution, not partial solutions.
- The agent of the blackboard model knows when it has discovered an optimal solution so that the agent can halt its processing. In the GA, the goodness of a solution is relative to other solutions in the population. It is possible for an agent to destroy its current optimal solution to investigate possibly better, but actually worse, solutions.
- Hints remain on the blackboard for easy access, so that in the blackboard model partial solutions are not destroyed. In the GA, memory is local to agents, and the amount of that local memory is limited by the population size. A partial solution can be lost due to selection (when there is epistasis) or to mutation.
- In the blackboard model, all of the agents have access to all of the hints. (However, the search for new information may in reality cost time.) In the GA, the copying of solutions is usually performed among subsets of agents as a result of the selection process. The sharing of partial solutions is performed among pairs of agents during crossover.

2.3 MULTIPLICATIVE PROCESSES AND THE LOGNORMAL DISTRIBUTION

The lognormal distribution has been used to describe processes in many fields, including: industry, economics, biology, ecology, and geology (Aitchison and Brown, 1976; Shimizu, 1988). For example, Mosimann and Campbell (Mosimann and Campbell, 1988) describe an active tissue growth model in which the tissue produced at prior time steps actively participates in the production of tissue at the current time step. Thus, if W_k is the amount of tissue at time k, then the growth ratio, $X_{k+1} = W_{k+1}/W_k$. Given W_0 is the initial amount of tissue, the tissue at time k is

$$W_k = W_0 \prod_{i=1}^{k} X_i, \quad k = 1, \ldots, n. \tag{1}$$

Unlike the inert tissue model, which is additive, the active growth model allows for both the growth and the decay of tissue at different times. Values of X_i greater than one indicate growth; values of X_i between 0 and 1 indicate decay.

The asymptotic distribution for a product of *random variable*s is generally lognormal. The name of the lognormal distribution is derived from the fact that a product can be transformed into a sum by taking the natural log of the terms. Because the sum of the log terms asymptotically converges to a Gaussian (Normal) distribution (as we know from the Central Limit Theorem), the limiting probability distribution of a product of random variables is usually described using the lognormal distribution. Given $x > 0$ in Eq. (1), the simple lognormal distribution is a function of a scaling parameter, μ, and a shape parameter, σ. For $-\infty < \mu < \infty$, and $\sigma > 0$, (Hahn and Shapiro, 1967, pp. 99) the two parameter lognormal is

$$f(x) = \frac{1}{\sigma x \sqrt{2\pi}} exp\left[-\frac{1}{2\sigma^2} (ln(x) - \mu)^2 \right]. \tag{2}$$

The three parameter version includes a threshold parameter, ε, which indicates the minimum value of x. Given $-\infty < \varepsilon < \infty$, $x > \varepsilon$, $-\infty < \gamma < \infty$, and $\eta > 0$ (Hahn and Shapiro, 1967, pp. 199), and

$$\eta = 1/\sigma \text{ and } \gamma = -\mu/\sigma$$

from the two parameter lognormal, the density of the three parameter lognormal is

$$f(x) = \frac{\eta}{(x-\varepsilon)\sqrt{2\pi}} exp\left[-\frac{\eta^2}{2} \left(\frac{\gamma}{\eta} + ln(x-\varepsilon) \right)^2 \right]. \tag{3}$$

The three parameter lognormal can be transformed to the Gaussian distribution as follows:

$$z = \gamma + \eta ln(x - \varepsilon). \tag{4}$$

If $\varepsilon = 0$, then the three parameter version of Eqn. (3) reduces to the two parameter version of Eq. (2).

2.4 MULTIPLICATIVE PROCESS IN COOPERATIVE LEARNING

The analysis of Clearwater et al. shows that the speedup in blackboard cooperative learning is due to the multiplicative aspect of the process. Their analysis is performed in two ways as briefly summarized below. The shape of the lognormal distribution shows that the characteristic acceleration of cooperative learning over individual learning. The long tail of the lognormal distribution represents an increase in the number of "superior performers" having fast search speeds.

2.4.1 Speed Distribution of Agents

Before examining multiplicative processes, let us start by examining the probability distribution of the time required for an agent to solve a problem individually using random search. Let us assume that an agent can look at a possible solution more than once (i.e., search with replacement). The probability distribution of an agent finding a solution at time t is

$$P(t) = \prod_{k=0}^{t-1} P_{failure}(k)\, P_{success}(t). \tag{5}$$

As Clearwater, et al. point out, if the probability of failure remains constant, then the resulting time distribution is a simple geometric. If the size of the search space is S, and the number of solutions is s, then the distribution becomes

$$f(t) = (1.0 - \frac{s}{S})^{t-1} (\frac{s}{S}). \tag{6}$$

If we change the replacement assumption so that the agent cannot reexamine a possible solution, then the fraction of failure s/S in Eq. (6) decreases as the search progresses due to a decrease in S. However, this decrease is initially small if the agent eliminates only one state in the search space at each step. Since there are still a large number of remaining potential solutions for the agent to examine, the distribution is still approximately geometric.

If we again change our assumptions so that each of the agents examines a changing blackboard for partial solutions, then $P_{failure}(t)$ in Eq. (5) becomes a random variable. The amount of reduction in the search space varies at each step as the product of the reduction at the previous time steps. Because Eq. (5) has the same form as Eq. (1), we can conclude that the time to solve the problem is lognormally distributed. If the time to solve the problem is lognormally distributed, then the speed of finding a solution (i.e., speed being the reciprocal of the time) is also lognormally distributed.

2.5 Examining the Distribution of Hint Values

Clearwater et al. (Clearwater, Huberman, and Hogg, 1991) also perform an analysis showing the multiplicative effect of hints in reducing the size of the search space. A lognormal distribution of hint values over a time interval results in a lognormal distribution in the speed of the agents finding a solution. The reduction in the size of the search space is expressed in terms of a recursive relationship as shown in Eq. (7). The equation includes the rate of individual working effort and the rate of cooperative sharing. The multiplicative effect of an additional j^{th} hint on the remaining size of the search space for the i^{th} agent is

$$d_i(j) = \left[d_i(j-1) - \frac{w_i}{q} \right] h_j, \tag{7}$$

where $d_i(j)$ is the resulting number of solutions that the agent has left to consider, h_j is the effectiveness of a hint, w_i is the rate at which the agent works on exploring possible solutions (i.e., states in the search space), and q is the rate of examining posted hints of

other agents. When h_j has a value less than one (and greater than zero), the j^{th} hint reduces the number of possible solutions that an agent must consider. The closer the hint value is to zero, the more beneficial is the hint. Hints given during the beginning of the search are usually more effective than those received later on. Based on Eq. (7), the reduction in the effective search space from the initial time zero after the $(j - 1)^{th}$ hint is

$$d_i(j) = d_i(0) \prod_{k=1}^{j} h_j - \frac{w_i}{q} \sum_{l=1}^{j} \prod_{k=l}^{j} h_l. \tag{8}$$

If there is very little individual search (i.e., $w_i \ll q$), then the hint values in Eq. (8) are nearly multiplicative. As a result, the distribution of hint values for a fixed amount of time (or number of hints) is lognormally distributed. This distribution can be easily transformed into a lognormal distribution of solution times. Since the distribution of hints effectively gives a distribution of the search space remaining, this distribution is proportional to the distribution of time remaining for the agents to find the solution. Thus, Clearwater et al. again demonstrate that the asymptotic probability distribution of the times for discovering a solution (as well as the speed) is lognormally distributed among the agents.

2.6 SCHEMA ANALYSIS AND MULTIPLICATIVE PROCESSES

The Schema Theorem also expresses a recursive relationship: the number m of schema H in the population from one generation to the next. If $p_H(s, t-1)$ is the probability that schema H survives after the disruptive effects of crossover and mutation, and if $f_H(t-1)/\bar{f}(t-1)$ is the proportion of schema H generated, then according to the Schema Theorem, the number of schema H in the next generation, $m_H(t)$, is bounded as shown in Eq. (9) (Holland, 1975):

$$m_H(t) \geq m_H(t-1)\,[p_H(s, t-1)] \left(\frac{f_H(t-1)}{\bar{f}(t-1)}\right). \tag{9}$$

After $t-1$ generations, the number of schema H is

$$m_H(t) \cong m_H(0) \prod_{k=0}^{t} \left(\frac{f_H(k)}{\bar{f}(k)}\right) [p_H(s, k)]\,. \tag{10}$$

GAs, when modified to perform function optimization, use fairly involved heuristics that go beyond the assumptions behind the Schema Theorem (e.g., single-point crossover, proportional selection).[3] These extended heuristics include the scaling of observed fitnesses, elitist strategies, ranked-based selection, various forms of crossover, adaptively changing crossover and mutation probabilities, etc. When performing function optimization, Eq.

3. De Jong points out the inappropriateness of using the Schema Theorem to analyze non-canonical GAs performing function optimization (De Jong, 1992).

(10) should probably be expressed in terms of some process-dependent heuristic, $\xi_H(k)$, which acts more like a random variable. Given

$$\xi_H(k) = \left(\frac{f_H(k)}{\bar{f}(k)}\right)[p_H(s,k)], \tag{11}$$

then

$$m_H(t) \cong m_H(0) \prod_{k=0}^{t} (\xi_H(k)) . \tag{12}$$

If the heuristic $\xi_H(t)$ increases the effectiveness of the evaluation of schema H over time, then the GA is multiplicative in terms of increasing or decreasing *a particular schema H*. If schema H is considered to be the optimal solution to the problem, and if the heuristic $\xi_H(t)$ is effective, then the GA's distribution of solution speeds over the population members might be described using some variation of the lognormal distribution.

Notice that Eq. (12) has a form similar to Eqn. (1) above. Schema H in this context refers to a partial solution. In effect, when $\xi_H(t)$ is between 0 and 1, it is analogous to the *value* of a hint in Eq. (8). As $\xi_H(t)$ gets closer to zero, the effectiveness of the "hint" in reducing schema H improves. Alternatively, a product greater than one is analogous to increasing a desirable schema H in the population. The individual agent's work effort is implemented through the mutation operator. If the individual work effort is small, then the effect on $\xi_H(t)$ is small.

The GA implements hint sharing through reproduction and crossover. The combined effect of reproduction and crossover can be thought of as regulating the rate of access to a hint on the blackboard. However, unlike the blackboard learner, this access rate is inseparable from the value of the hints in Eqn. (12). The GA must use its population not only to investigate solutions, but it must also use this finite memory capacity to reflect the quality of hints within the population. The GA increases its memory allocation to better performing schema in order to increase the rate of access to better hint values. This dual use of memory means that a desirable schema H may initially go through several short cycles of growth and decay.

3 AN EMPIRICAL LOOK AT THE PROBLEM

The next phase of this study is to explore GA behavior empirically using a few test cases to see if the GA's speed distribution for finding a solution compares favorably to the lognormal distribution. In order to obtain a general estimate of the speed of finding solutions, the GA runs until all of the agents have found the optimal solution at least once. During the run, the time that an agent first discovers the optimum is recorded, even though the agent's answer may change subsequently due to crossover and mutation. Since the information is still within the population, for the most part, an agent is credited with success the first time it encounters the solution.

All of the experiments reported here use modifications of GENESIS-4.5 (Grefenstette, 1983). The modified GA treats solutions and agents separately during reproduction and crossover. The shuffling of population members to assure random mating means that agents, and not their solutions, are rearranged. In this way, it is possible to track when an agent first finds an optimal solution.[4]

This study examines three optimization problems: functions F3 and F5 from De Jong's test suite (De Jong, 1975), and the test function F7, studied by Schaffer et al. (Schaffer, Caruana, Eshelman, and Das, 1989). Clearwater et al. solve examples of cryptarithmetic problems using a blackboard cooperative learning system to test their theory empirically and to provide a quantitative evaluation of the value of cooperation in problem solving. The cryptarithmetic problem is an example of optimization under constraints. For purposes of comparison, this study also applies a GA to solve one of the more difficult of the cryptarithmetic problems reported by Clearwater et al.

This study represents the beginning a larger endeavor. The work starts by examining a few functions using informal graphical methods to determine the reasonableness of applying the lognormal model. When appropriate, the empirical data are fitted to the three parameter lognormal by using an inverse transformation technique in which distribution parameters are estimated from percentiles of the actual data. The data are transformed to the Gaussian distribution using Eq. (4), and then a Kolmogorov-Smirnoff goodness-of-fit test is performed between the data and a three parameter lognormal distribution having the same parameter values.[5]

---

4. Normally, in GENESIS-4.5, each generation is treated separately, since new structures represent the children of parent structures and not new solutions.

5. Goodness-of-fit techniques differ from usual statistical techniques (D'Agostino and Stephens, 1986). Usually, statistical experiments are designed to disprove a null hypothesis so that an alternative hypothesis can be accepted. In goodness-of-fit testing, the opposite approach is used: an effort is made to confirm the null hypothesis. The alternative hypothesis is often a composite hypothesis, giving little or no information about the distribution. As a result, proving the goodness of a fit is not usually definitive. Based on some understanding of the underlying process, a set of null hypothesis are tested; in turn, each of these null hypotheses are examined using several goodness-of-fit techniques.

4 EXAMINING SOME GA OPTIMIZATION PROBLEMS

In this study we are interested in emphasizing convergence or improvement in off-line performance, because we would like all of the agents to find the optimal solution. Very often, in order to achieve convergence, the GA needs a large population and a small mutation rate. So all of the optimization runs use a population size of 100 or 200 and mutation rate of 0.001. If we were more interested in on-line performance, a smaller population and a higher mutation rate would be desirable (Schaffer, Caruana, Eshelman, and Das, 1989). For F7, the most difficult problem of the three optimization problems, an elitist strategy is used instead of a higher mutation rate. All of the optimization runs use a scaling window of five generations.

Figure 1 (a) shows a graphic plot of the speed points for a run of F3. If a transformation of the log of the data to the Gaussian distribution is correct, the data points should lie along a straight line. Figure 1 (b) shows the corresponding fit to the empirical speed distribution among the agents. This fit passes the Kolmogorov-Smirnoff test at the 95% level of confidence.[6] Smoothing bin counts with neighboring bins provides an even better fit (not shown). Figure 1(c) show the growth rate of all schemata mapping into the optimal solution. During the beginning of the search, the optimal schemata exhibit short bursts of growth and then appear to decay (a growth ratio less than one) in a subsequent generation. The pattern of growth burst followed by decay, proceeds in a cyclic fashion until the growth rate begins to level out. Optimal schemata temporarily disappear (intact) while the GA tests other non-optimal schemata. When the optimal schemata reappear, the large burst of growth compensates for the temporary hiatus of the schemata.

Figures 2(a) and 2(b) shows the results of a comparable run using function F5. The nonlinearity in Figure 2(a) indicates a shorter tail than what would be expected for the lognormal. The fit shown in Figure 2(b) passes the Kolmogorov-Smirnoff test at the 95% confidence level.

Figure 3 shows similar plots for function F7. By incorporating the elitist strategy, a large percentage of the agents in the population discover the optimal solution fairly quickly. The empirical distribution in this case is actually negatively skewed. The elitist strategy dominates the other learning heuristics. The result is an unidentifiable mixed distribution.

---

6. The Kolmogorov-Smirnoff test examines the general shape of the distribution and not expected bin counts. The bin counts in these experiments are too small to perform Chi-Square tests.

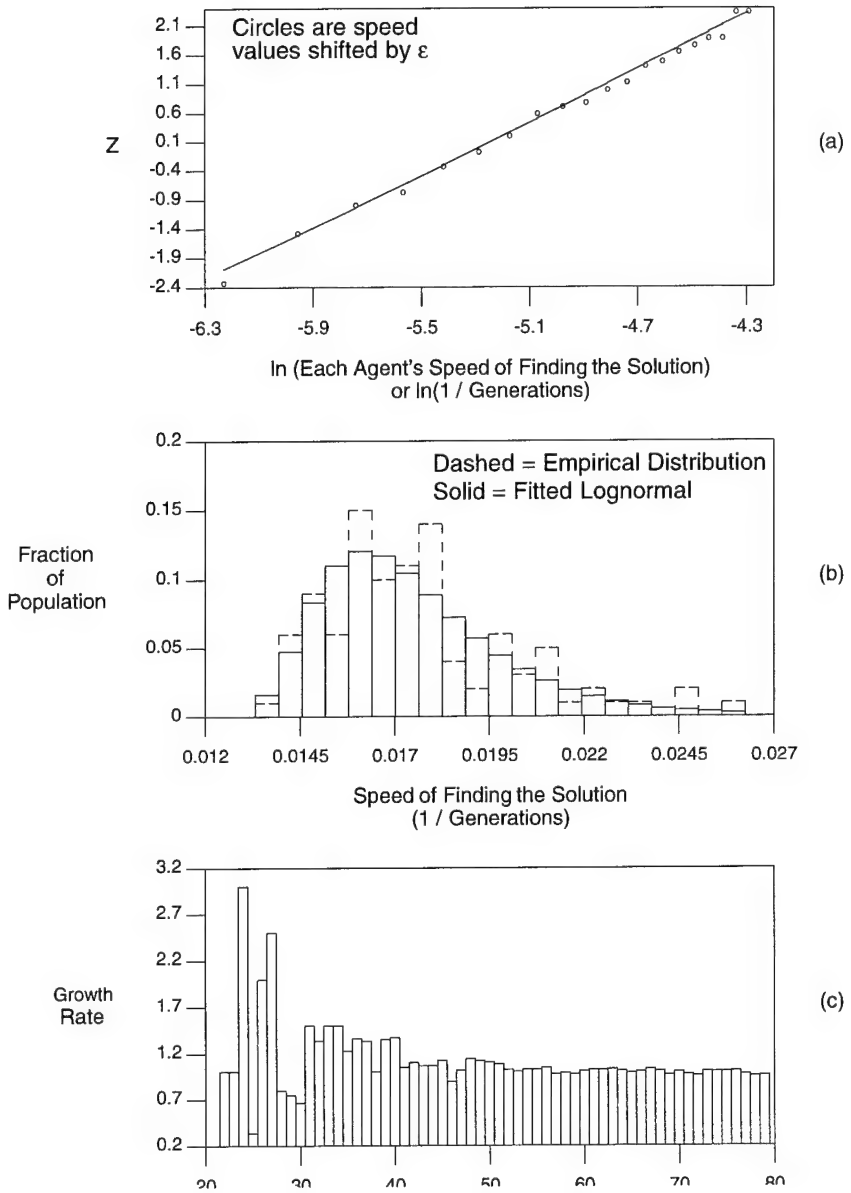

Figure 1: Test Results for De Jong's F3 Function

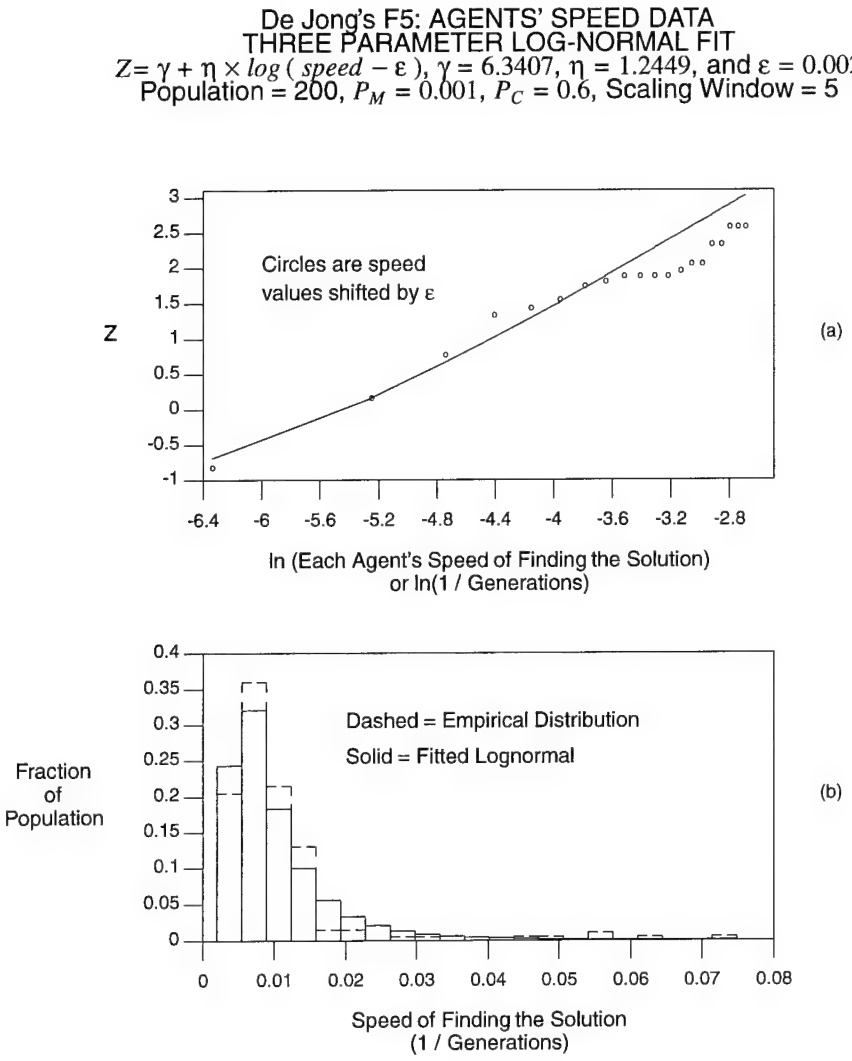

Figure 2: Test Results for De Jong's F5 Function

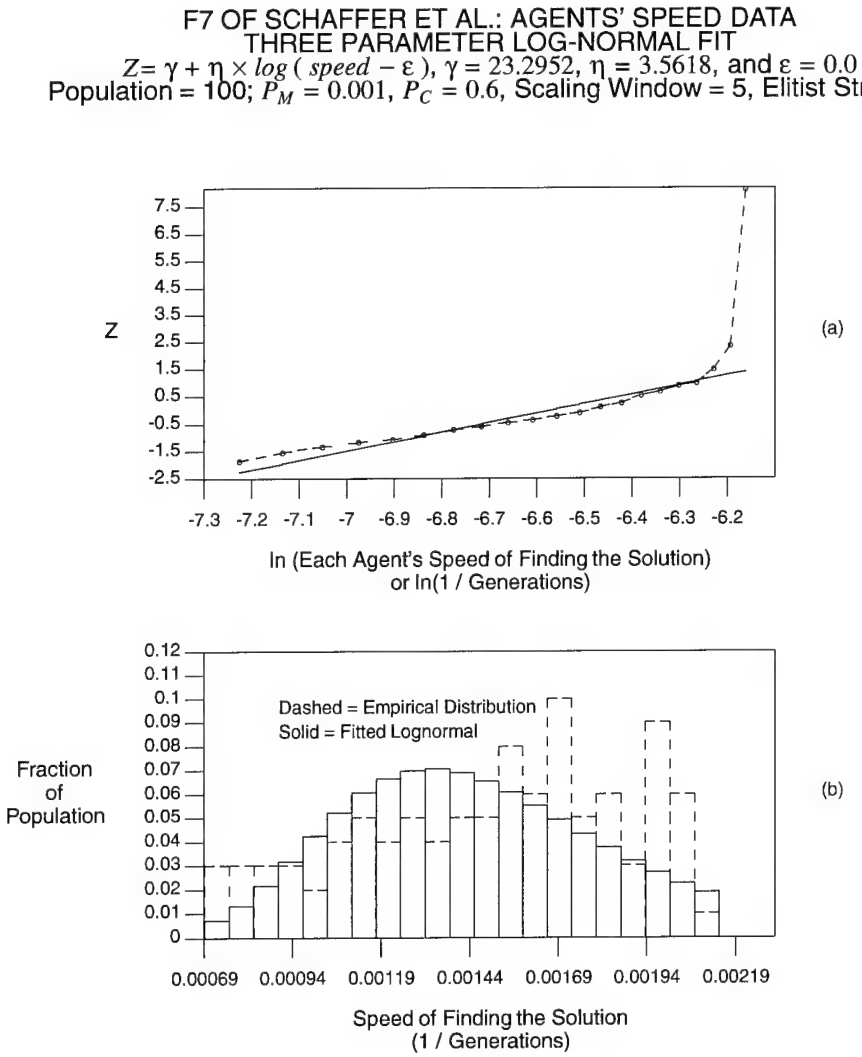

Figure 3: Test Results for Function F7

4.1 THE CRYPARITHMETIC PROBLEM

In cryptarithmetic, an addition problem is expressed using letters. The objective of problem is to assign a *unique* numeral to each letter used so that the addition works correctly. For example, one problem explored by Clearwater et al. is DONALD + GERALD = ROBERT. In this problem, there is one solution and 3,628,800 legal combinations of decimal digits in the initial search space. The solution is A = 4, B = 3, D = 5, E = 9, G = 1, L = 8, N = 6, O = 2, R = 7, and T = 0.

A GA population structure consists of the assigned value for each of the ten letters; i.e., an allele is a decimal digit. For example, given the fixed letter sequence {A B D E G L O R T}, the correct assignment of the digits is {4 3 5 9 1 8 6 2 7 0}. In order to ensure that only legal representations are maintained in the population, the GA employs Syswerda's swap mutation, and a slight modification of his position-based crossover (Syswerda, 1991), which is called here, for convenience, *assignment crossover.* Assignment crossover ensures that the absolute allele assignments of the parents are maintained in the children as much as possible (not just relative positions). The assignment crossover proceeds in three steps: (1) exchange allele information at selected positions; (2) copy the alleles of a parent into the empty positions of the corresponding child (e.g., from parent 1 to child 1), unless the child just inherited the value from the other parent, and (3) place the crossed over values of a parent (e.g., alleles from parent 1 to child 2) into the remaining empty slots of its own child, if possible. Four positions are randomly selected for application of the operator.

Parent 1:	0	1	3	7	9	2	5	6	8	4
Parent 2:	7	3	5	9	8	4	0	2	6	1
Selected Positions:	*		*				*	*		
Child 1:	7	1	5	3	9	6	0	2	8	4
Child 2:	0	7	3	9	8	4	5	6	2	1

As a result of the assignment crossover above, parents and their corresponding children have the values 1, 9, 8, and 4 in the same positions. This crossover operator minimizes the disruption in the assignment of the digits.

The evaluation function in the GA computes the sum of DONALD + GERALD = ROBERT based on the current assignment of the letter values. For each column added correctly, the agent receives partial credit depending on the number of letters participating in the sum. For example, if in adding the right-most column, i.e., D + D = T, the agent were to get the correct answer, then the partial evaluation would be 2; if the agent were also to add the left most column correctly (after taking care of any carry, of course), then the agent would receive an additional 3 points. The totally correct answer is worth 14 points. Unlike most traditional GA problems, this problem explicitly uses partial evaluations to compute the overall fitness of an agent's trial solution. Nevertheless, population members only receive evaluations of entire solutions.

The mutation rate used for this problem is higher than the rate used in the preceding optimization problems ($p_m = 0.1$ instead of $p_m = 0.001$) because each letter can take on ten values instead of two. Without the use of a high mutation rate, the GA prematurely converges to a suboptimal solution. A small crossover rate (around 0.2) is used in conjunction with the high mutation rate. The GA also tends to prematurely converge when the crossover rate exceeds 0.21.

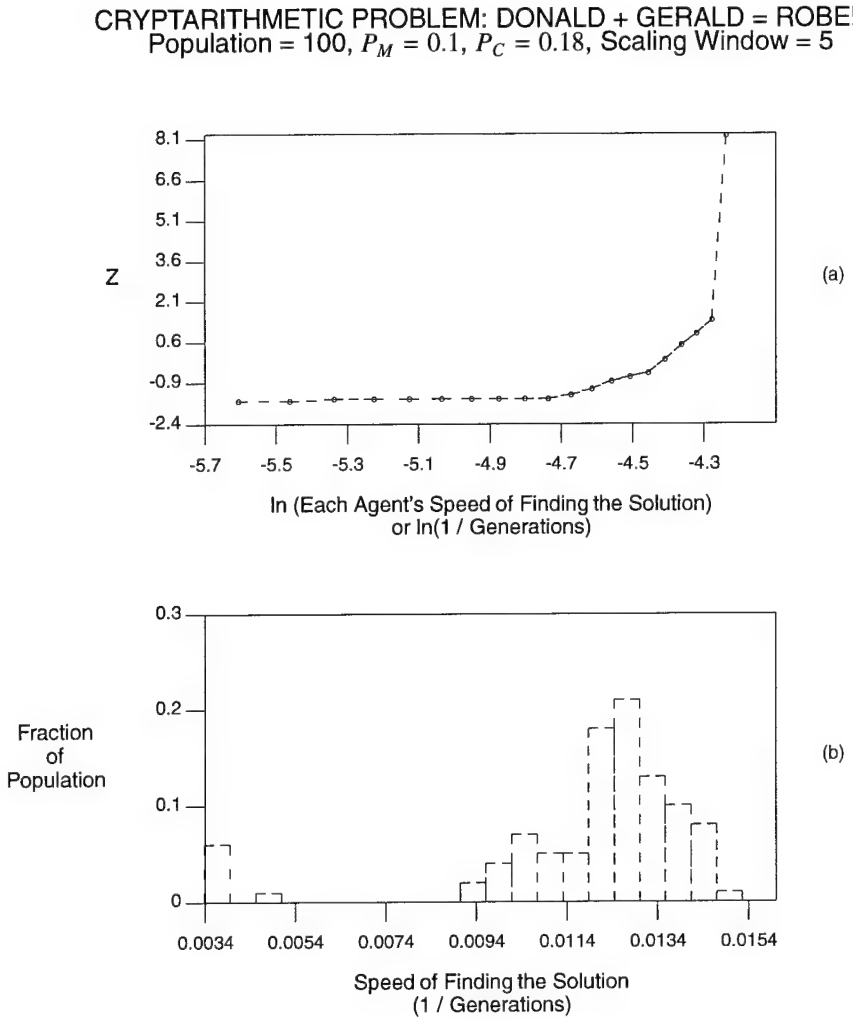

Figure 4: Test Results for the Cryptarithmetic Problem

Figure 4 shows the result of one of the runs. In Figure 4(a), the piecewise linear appearance of the plot suggests that for pieces having a non-zero slope, the distribution is a mixture of lognormal distributions. Slightly curved shapes represent transitions between the distributions. The disruptive effect of mutation prevents a more continuous growth in the solutions.

Figure 5 shows the result of having a varying number of crossover points, ranging from one to nine. There are essentially two separate intervals where the agents discover the solution. The additional disruptive effect of the crossover prevents the continued growth of

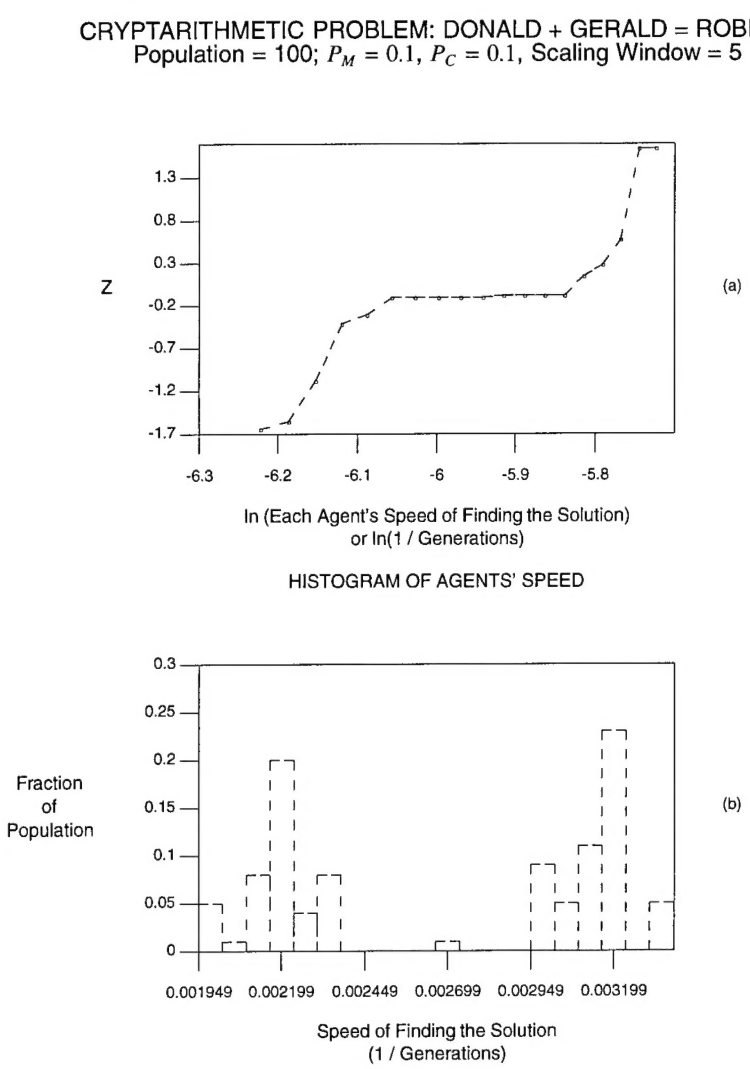

Figure 5: Additional Test Results for the Cryptarithmetic Problem

the solution starting around generation 310 so that the remaining agents do not discover the solution until around generation 450.

5 CONCLUSIONS

Viewing the GA as a cooperative learner provides additional insight into the function of the algorithm. This paper begins to explore the GA as a cooperative process. Due to information sharing among members of the population, the GA can be described metaphorically using a cooperative learning framework.

The actual distribution of solution speeds for the GA depends on many factors. The shape of the distribution is sensitive to parameter settings which affect the balance between cooperation and exploration in the GA. In addition, the propagation of solutions through the population using the mechanisms of reproduction and crossover affect the shape of the speed distribution. The GA increases its memory allocation to better performing schemata in order to increase the rate of access to those schemata; however, the same memory space is used to test potential solutions in the search space. Because the GA uses its memory for both purposes, a relatively fit schema may go through short cycles of growth and decay toward the beginning of the search. In effect, a schema needs to time-share the population memory with other schema. Nevertheless, the overall speed distribution among the agents for finding a solution may be approximately lognormal in some cases.

Future research will address the cooperative aspect of the GA by examining: (1) more test problems; (2) different goodness-of-fit tests; (3) other heuristics used by the GA (e.g., rank based selection), and (4) modifications of the GA. For example, it would be interesting to test what effect, if any, increasing the storage capacity of each agent in the population has on the speed distribution of agents. It might also be interesting to study the effect of having crossover among more than two parents as a way of increasing the communication in the population.

Acknowledgments

I would like to thank the FAW Institute for inviting me to their 1991 "Adaptive Learning and Neural Networks" workshop in Ulm, Germany. It was at this workshop that I first heard Bernardo Huberman speak about his research on cooperative learning.

References

David H. Ackley (1991). Re: GA's and Hill-Climbers. *GA-List, Vol 5, Issue 23* (A moderated bulletin board).

J. Aitchison and J. A. C. Brown (1976). *The Lognormal Distribution*. London, Eng.: Cambridge University Press.

Alan H. Bond and Les Gasser (Eds) (1988). *Readings in Distributed Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann.

Scott H. Clearwater, Bernardo A. Huberman, and Tad Hogg (1991). Problem Solving by Committee. Presented at the FAW workshop on Adaptive Learning and NNs, July 2 - 7, in Ulm, Germany.

Jeffery A. Clouse and Paul E. Utgoff (1992) A Teaching Method for Reinforcement Learning. *Machine Learning: Proceeds of the Ninth International Workshop (ML92)*, pp. 92 - 101, Derek Sleeman and Peter Edwards, eds., San Mateo, CA: Morgan Kaufman.

Kenneth A. De Jong (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Doctoral Dissertation, Univ. of Michigan.

Kenneth A. De Jong (1992). Genetic Algorithms are NOT Function Optimizers. In *Foundations of Genetic Algorithms 2*, D. Whitley (Ed.), San Mateo: Morgan Kaufmann.

Richard D. Fennell and Victor R. Lesser (1989). Parallelism in Artificial Intelligence Problem Solving: A Case Study of Hearsay II. *Readings in Distributed Artificial Intelligence*, pp. 106 - 119, Alan H. Bond and Less Gasser (eds.), San Mateo, CA: Morgan Kaufman.

John J. Grefenstette (1983). *A User's Guide to Genesis.* Technical Report CS-83-11, Computer Science Department, Vanderbuilt University.

David E. Goldberg (1989). *Genetic Algorithms in Search, Optimization & Machine Learning. Reading*, MA: Addison-Wesley.

Gerald J. Hahn and Samuel S. Shapiro (1967). *Statistical Models in Engineering.* New York: John Wiley and Sons, Inc.

John Holland (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: The University of Michigan Press.

B. A. Huberman and T. Hogg (1987). Phase transitions in artificial intelligence systems. *Artificial Intelligence*, 33:155-171.

Bernardo A. Huberman (1990). The Performance of Cooperative Processes. *Physica D* 42, 38-37.

William A. Kornfeld (1981). The use of parallelism to implement heuristic search. Technical report, Massachusetts Institute of Technology Artificial Intelligence Laboratory, 1981. AI. Memo No. 627.

James E. Mosimann and Gregory Campbell (1988) Applications in Biology: Simple Growth Models. In Edwin L. Crow and Kunio Shimizu (Eds.). *Lognormal Distributions: Theory and Applications*, NY: Marcel Dekker.

J. David Schaffer, Richard A. Caruana, Larry J. Eshelman, Rajarshi Das (1989). A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. *Proceedings of the Third Inter. Conf on Genetic Algorithms.* San Mateo, CA: Morgan Kaufman.

Kunio Shimizu (1988). History, Genesis, and Properties. In Edwin L. Crow and Kunio Shimizu (Eds.), *Lognormal Distributions: Theory and Applications,* NY: Marcel Dekker.

Reid G. Smith and Randall Davis (1989). Frameworks for Cooperation in Distributed Problem Solving. *Readings in Distributed Artificial Intelligence,* pp. 61 -70. Alan H. Bond and Les Gasser (Eds). San Mateo, CA: Morgan Kaufmann.

Gilbert Syswerda (1991). Schedule Optimization Using Genetic Algorithms. In *Handbook of Genetic Algorithms.* Lawrence Davis (Ed), NY, NY: Van Nostrand Reinhold.